dSPACE and Real-Time Interface in Simulink

Azad Ghaffari

SAN DIEGO STATE UNIVERSITY

San Diego State University
Department of ECE
San Diego
CA 92182-1309
12/20/2012

This document provides a tutorial introduction to the dSPACE software (ControlDesk Next Generation version 4.2.1), the dSPACE DS1104 R&D controller board, and their use in development and implementation of maximum power point tracking (MPPT) for a single photovoltaic (PV) module using extremum seeking (ES) in Simulink software. It is intended for use as a quick-start guide to dSPACE hardware/software for a university course. Full details on the dSPACE hardware and software can be found in the dSPACE documentation. This presentation is prepared based on the following package: MATLAB Version 7.12(R2011a), Simulink Version 7.7 (R2011a), and dSPACE DVD Release 7.3 (2012).

1 dSPACE and Real-Time Interface in Simulink

Contents

1. System Requirements

You can use an x86-compatible personal computer as a host PC for your dSPACE applications with following specifications:

Host processor:	Pentium 4 at 2 GHz (or equivalent)
Main memory:	2 GB RAM or more (recommended)
Disk space:	5.5 GB on the program partition for complete installation of the DVD
Dongle licenses:	A USB port: To install the execution key (dongle)
Required slots:	To install a DS1104, you need one free 33 MHz/32-bit 5 V PCI slot

ControlDesk Next Generation version 4.2.1 which is a part of dSPACE DVD Release 7.3 supports following operating system:

Windows XP Professional (32-bit version) with Service Pack 3
Windows Vista Business, Ultimate, and Enterprise (32-bit version) with Service Pack 2
Windows 7 Professional, Ultimate, and Enterprise (32-bit or 64-bit versions) with Service Pack 1

64-bit MATLAB versions are not supported. Real-Time Interface to Simulink which is a part of "RCP and HIL software" (Rapid Control Prototyping and Hardware-in the-Loop software) supports the following versions of MATLAB: R2012a, R2011b, R2011a, R2010bSP1, R2010a, R2009bSP1.

2. dSPACE Package

To implement a real-time control loop using dSPACE and MATLAB we need following items.

1. dSPACE DS1104 R&D Controller Board

2. Dongle licenses on a USB flash disk

3. License.dsp file
4. Keys.dsp file
5. Connector panel CP1104

3. Real-Time and the Structure of a Real-Time Program

Suppose we have a continuous system and we want to control it with a discrete controller which has sampling time period of T. The following figure shows the connections between the system and its controller. We need analog-to-digital converters (ADC) to read the information of the sensors. Also to apply the control commands we need digital-to-analog converters (DAC).

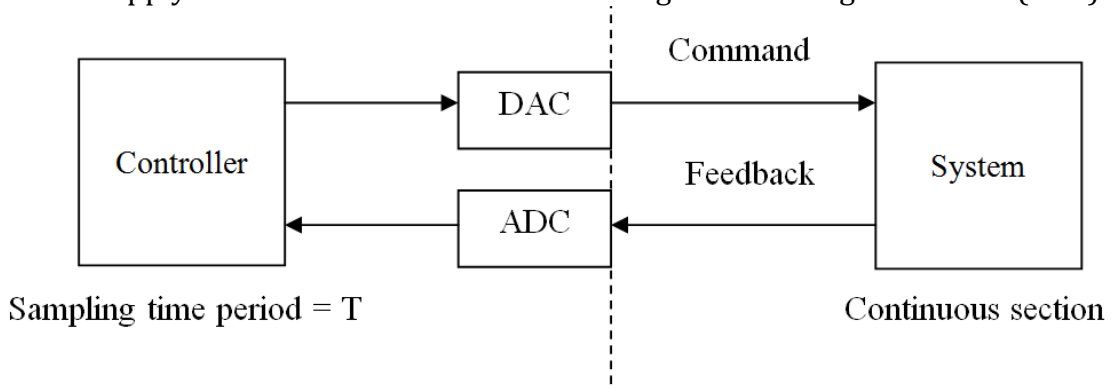


Fig. 1: Real-time control structure

Because this system or object has certain dynamics associated with it, you have to control it based on those dynamics. Therefore we say that the physical system will have a time constant, from which you will derive a step size or sample time for your control program. The challenge is to not only use that sample time in the numerical calculations that make up your control algorithm, but also to execute that algorithm within that sample time. You have to start each "step" of your program exactly one sample time or step size apart, and thus have to finish the

computation of each step within the sample time, i.e. before the next step starts. This is real-time. Please see the diagram below.

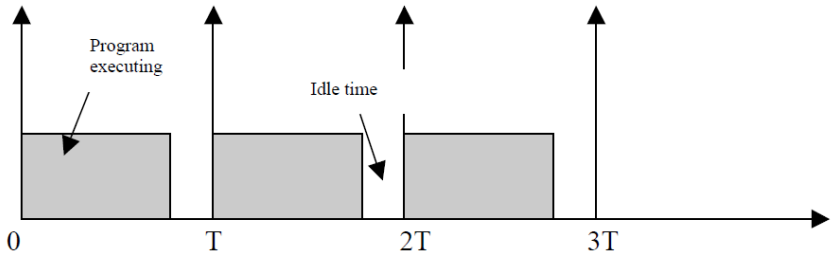


Fig.2: Real-time control timing

If the sample time of our program is T, you can see that the program is executed at distinct points in time that are one sample time apart. You will also note that each step of the program finishes executing before the next step is due to start; thus this program is running in real-time. If however the computational demands of the program cause the processor to take more time than the sample time then we have an overrun condition, and our program cannot run in real-time.
The overall structure of a real-time program can be simplified for explanation purposes into three main sections: Initialization, the real-time task or tasks, and the background. The initialization section is code that is executed only once at the start of execution, upon download of the program. In this section you will have functions that, for initialization of the system, are only needed to run once. The next part of the program is the real-time part, the task, represented by the gray sections in the diagram above. This is what is executed periodically based on the sample time. This part is the heart of the control program; for this, you read inputs (e.g., from an ADC), compute your control signals, and write outputs (e.g., with a DAC). Note that depending on what your control application is you may have multiple tasks in your model. Finally, the last section is the background; this is code executed in the "idle" time between the end of computation of a step and the start of the next step.

4. Photovoltaic Module and Maximum Power Point Tracking

The PV cell is modeled as an ideal current source of value i_s in parallel with an ideal diode with voltage v_d. Electrical losses and contactor resistance are accounted for by the inclusion of the parallel and series resistances R_s and R_p, respectively. The amount of generated current i_s is dependent on the solar irradiance S and the temperature T.

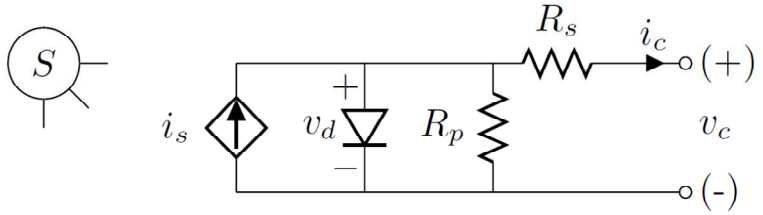


Fig. 3: PV module electrical equivalent circuit

As is clear from following figure the power-voltage (P–V) characteristic has a unique but (T, S) dependent peak.

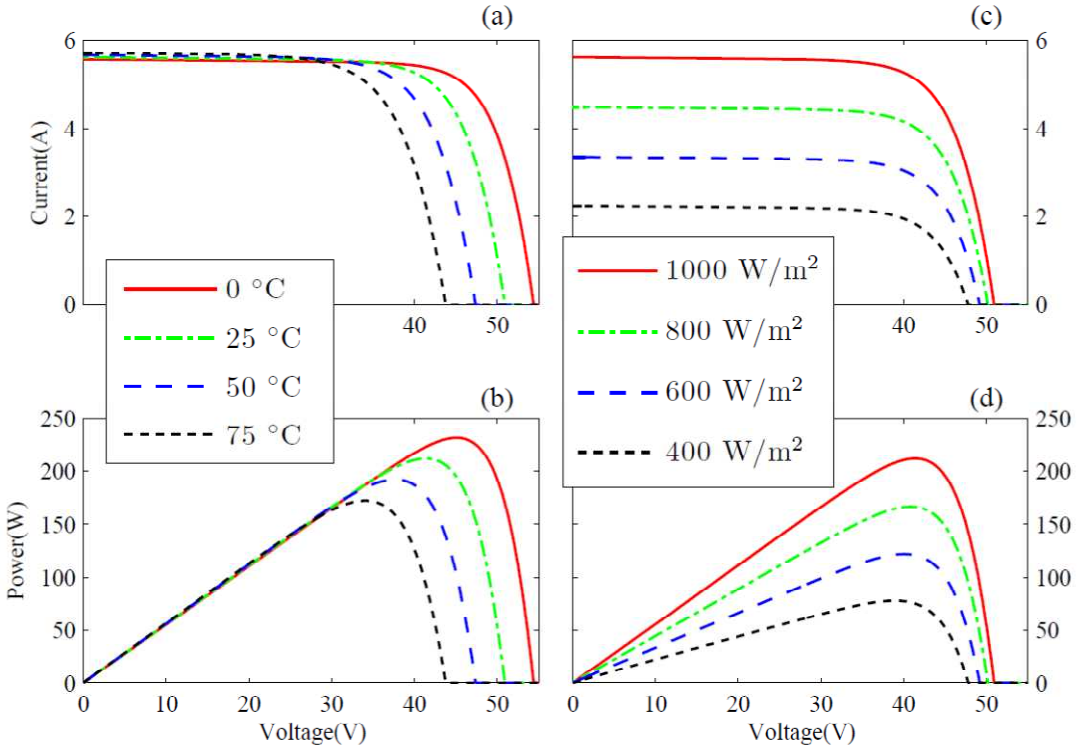


Fig. 4: Power and current variations of a PV module for different solar irradiance an environmental temperature

It is the job of the MPPT algorithm to automatically track this peak. In many grid-tied PV systems (including our current work), this is done by means of a separate DC/DC power electronics stage. Here we use a DC/DC buck converter as follows.

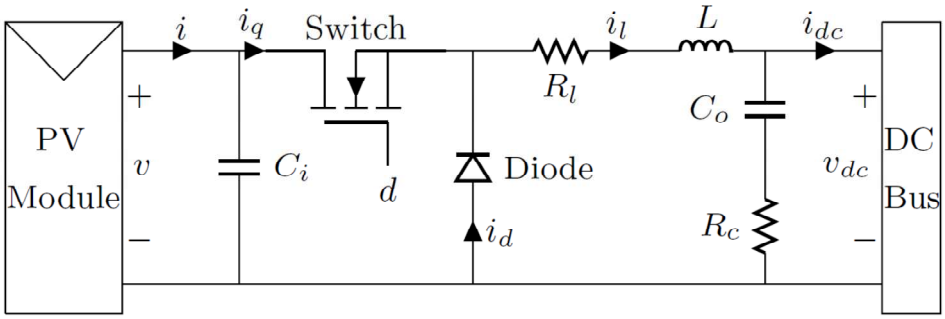


Fig. 5: DC-DC Buck converter to harvest power from a PV module

The averaged model of the buck converter in Continuous Conduction Mode (CCM) is described as

$$v = \frac{v_{dc}}{d}$$

where d is the pulse duration applied from pulse-width modulation unit to the gate of the switch. Variation of the power versus pulse duration for a PV module with $P_m = 12W$, $V_{oc} = 21.6$ V, $I_{sc} = 800$ mA, $V_m = 17.2V$, and $I_m = 700mA$ under standard test conditions is shown in the next figure.

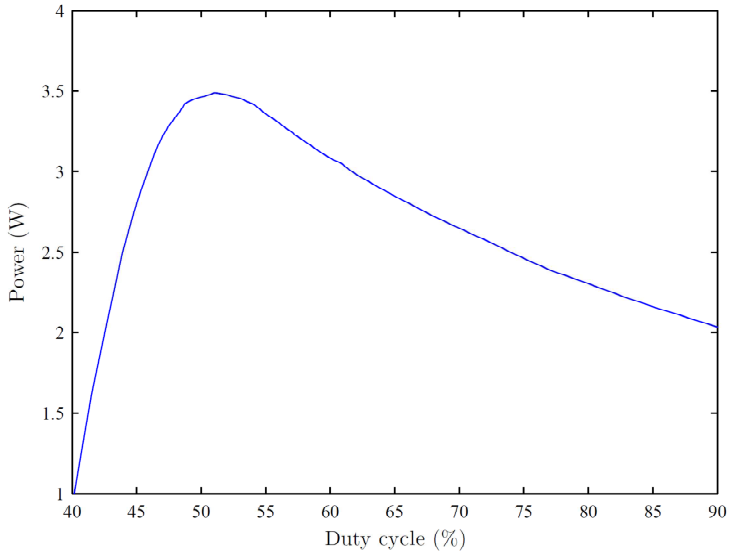


Fig. 6: Power versus duty cycle

When power is less than maximum value and the duty cycle is less than optimal duty cycle the curve has a positive slope and increasing the pulse duration results in higher power generation. When the duty cycle is larger than the optimal duty cycle the power curve has a negative cure and decreasing the pulse duration generates more power. At the peak point the slope of the curve is zero and there is no need to change the pulse duration. Based on this information we employ extremum seeking algorithm to estimate the gradient of the cost function and to implement the gradient descent optimization scheme. The proposed scheme is shown following.

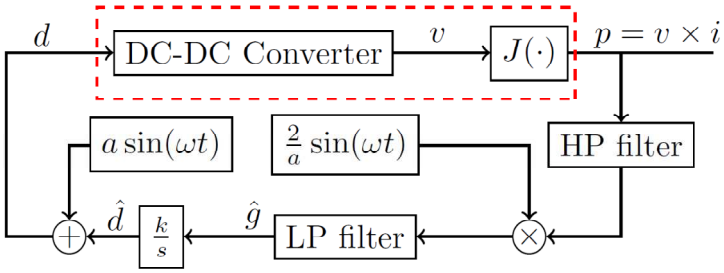


Fig. 7: Extremum seeking algorithm for Maximum Power Point Tracking of a PV module

Suppose we have the estimate of the pulse duration, $\hat{d}$. If this value is less than optimal duty cycle, the power varies in phase with the perturbation input. If the estimate of pulse duration is greater than the optimal duty cycle the power change is out of phase with the perturbation input. This causes the estimate of the gradient, $\hat{g}$, be positive or negative, respectively. The high-pass filter removes the DC part of the power and the low-pass filter is used to remove the oscillatory

parts of the estimate of the gradient. We use the Power-pole circuit board to construct the DC/DC buck converter.

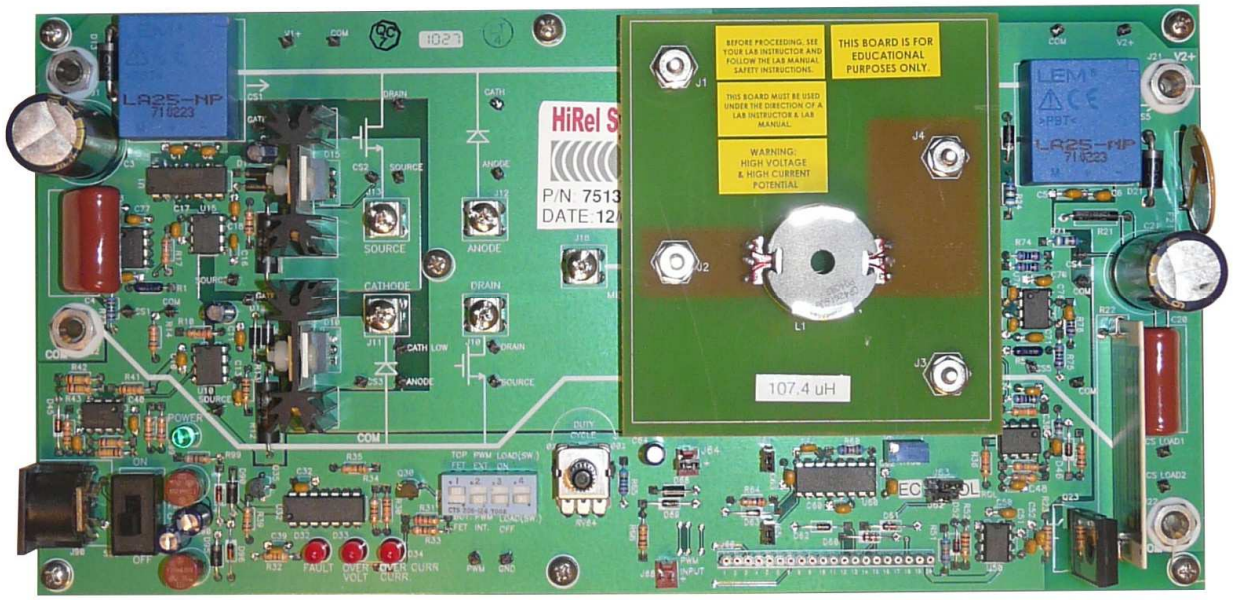
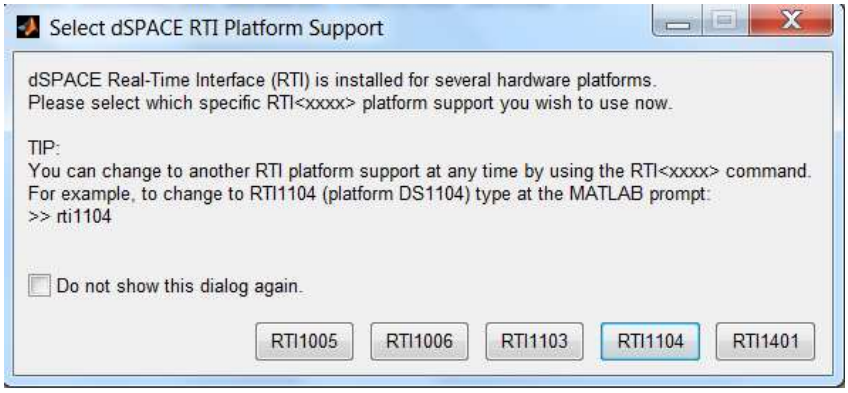


Fig. 8: Power-pole board used as a buck converter.

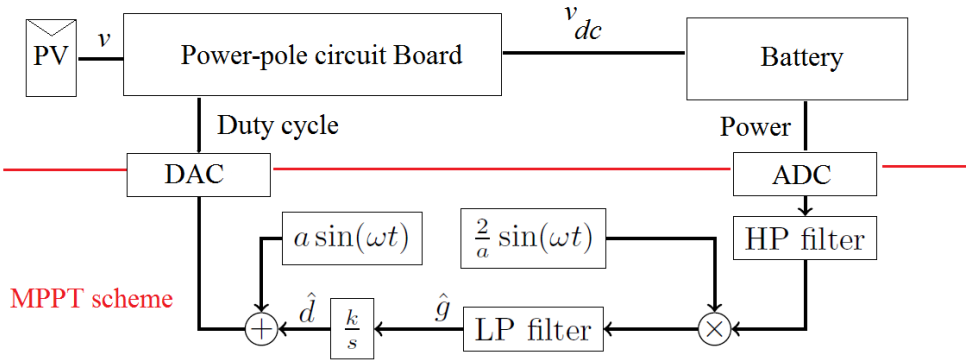
5. Controller Design and Implementation in Simulink

In this section we will discuss how to use Simulink for controller design and how to compile the Simulink model into code that will run on the dSPACE board for real-time implementation of the controller. When we start MATLAB following message appears, which says that dSPACE Real-Time Interface (RTI) is installed for several hardware platforms, in this case DS1104. To stop showing this message when MATLAB starts you can check the box.

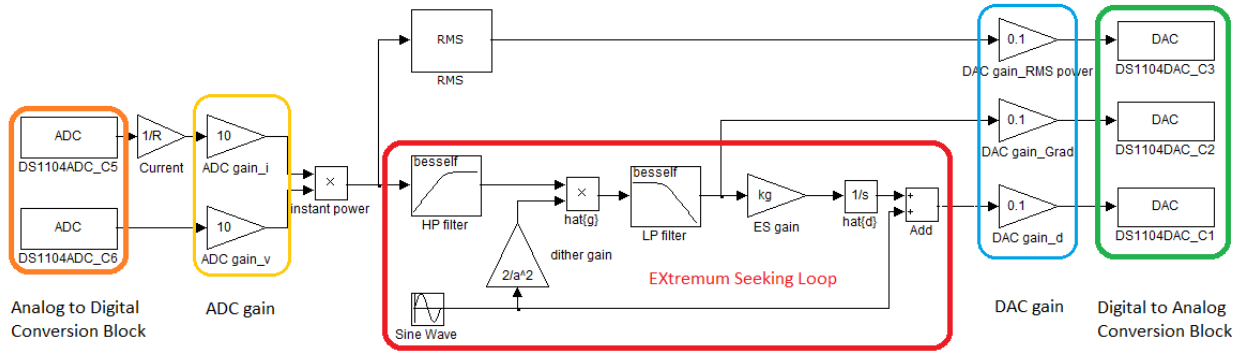


The closed-loop system is shown as follows. We need to measure the power generated by the PV module. For this purpose we measure the current and voltage of the DC bus using the ADC inputs

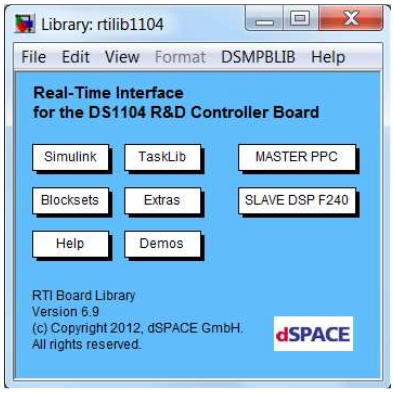
of DS1104. The command generated by the extremum seeking is the pulse duration which applies to the input of the PWM generator.



Suppose that you build a maximum power point tracking based on extremum seeking in Simulink as shown below. Save your project in *"E:\Work"*. Now that we have the signals that we need to sense, current and voltage, and actuate, duty cycle, we can consider the development of the Simulink model of the controller shown below:

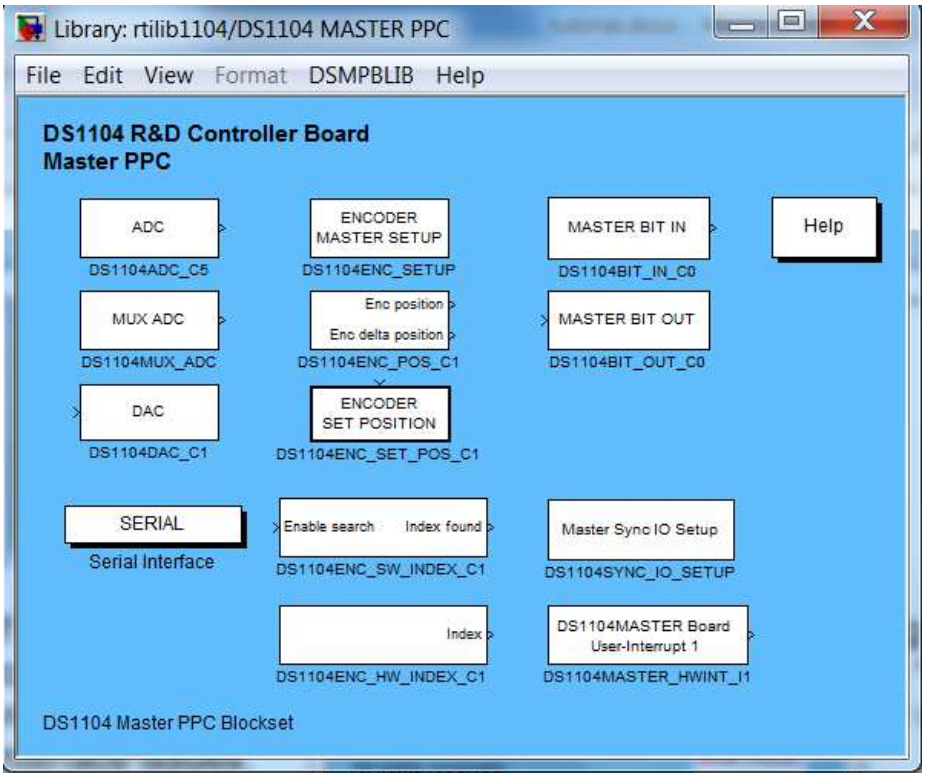


The construction of this block diagram will be discussed in more detail below. For now, focus on how we create the software interface between the controller and the plant (i.e., the interface that generates control inputs and read sensor values). The digital to analog conversion (DAC) blocks are provided in Simulink when the dSPACE software is available. Hence, we use a DAC block as shown above to generate the control input to the plant and an ADC block to read the signal. To see the dSPACE blocks one can type `rti` from the MATLAB command window. If you do that the following window is shown:



If you double-click on each of these blocks, you are going to find the blocks necessary to build the simulation that you need. Note that there are *"Demos"* that may be useful to you. Also, note that there is a *"Help"* button you may find useful. Next, we will discuss interface issues.

The RTI1104 Board Library seen above is divided into some main sections. The I/O resources of the DS1104 are split between the two processors on the board, the Master PPC (Power PC) and the Slave DSP F240. By clicking on either one you will have access to blocks you can place in your model that provide I/O functionality associated with the respective processor. For this tutorial we will focus on the group of blocks contained in the Master PPC section. If you double-click on this you will get the following window:



As you see, this window has some of the most commonly used elements for the controller board, such as ADCs, DACs, Encoders, etc. If you double-click on any of these I/O blocks you will get its

respective configuration dialog box, and one of the buttons you will see in this dialog box is *"Help"*. Clicking on this will launch the dSPACE HelpDesk exactly at the page referencing that particular block. Here, we clicked on dSPACE Help and downloaded the relevant information on the ADC and DAC that we need for the temperature control problem. You can also launch the dSPACE HelpDesk from the *''Start>All Programs>dSPACE ControlDesk 4.2.1>dSPACE HelpDesk (ControlDesk 4.2.1)"*, or if you are using ControlDesk NG you can launch it from the Help menu or simply by hitting the *"F1"* key.

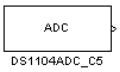
5.1 Analog to Digital Conversion (ADC) and Signal Scaling

The master PPC on the DS1104 controls an ADC unit featuring two different types of A/D converters:

- One A/D converter (ADC1) multiplexed to four channels (signals ADCH1 ... ADCH4).The input signals of the converter are selected by a 4:1 input multiplexer. The A/D converters have the following characteristics:
  - 16-bit resolution
  - ±10 V input voltage range
  - ± 5 mV offset error
  - ± 0.25% gain error
  - >80 dB (at 10 kHz) signal-to-noise ratio (SNR)
- Four parallel A/D converters (ADC2 ... ADC5) with one channel each (signals ADCH5 ... ADCH8). The A/D converters have the following characteristics:
  - 12-bit resolution
  - ±10 V input voltage range
  - ± 5 mV offset error
  - ± 0.5% gain error
  - > 70 dB signal-to-noise ratio (SNR)

To configure the software so that it can get this signal into the controller we click on *"ADC"* in the upper left corner (note the label on the bottom of that button). In the window that comes up there is a Help button. If you click it, you will see:

DS1104ADC_Cx



Purpose

To read from a single channel of one of 4 parallel A/D converter channels.

I/O mapping

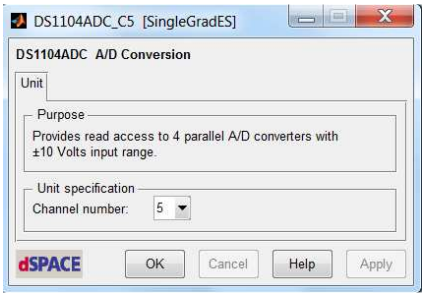
For information on the I/O mapping, refer to ADC Unit.

I/O characteristics

Scaling between the analog input voltage and the output of the block:

Input Voltage Range	Simulink Output
−10 V ... +10 V	−1 ... +1 (double)

Here, when you place an ADC block in a Simulink model (by drag and drop) and then double click it, all you need to select is the *"Channel number".* Next, it is important to understand the *"scaling"* that occurs in acquiring the signal. The physical input signal input range is –10V to +10V. dSPACE always scales this by a factor of 0.1 (multiplies by this number) to place the value on a range of –1V to +1V. We need to take the ADC signal and multiply by 10 to remove the scale factor.



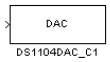
5.2 Digital to Analog Conversion (DAC) and Initialization /Termination

The master PPC on the DS1104 controls a D/A converter. It has the following characteristics:

- 8 parallel DAC channels (signals DACH1 … DACH8)
- 16-bit resolution
- ±10 V output voltage range
- ± 1 mV offset error, 10 □V/K offset drift
- ± 0.1% gain error, 25 ppm/K gain drift
- >80 dB (at 10 kHz) signal-to-noise ratio (SNR)
- Transparent and latched mode

To configure the software to generate the output signals we click on *"DAC"* on the left side, third block down (note the label on the bottom of that button). In the window that comes up there is a Help button. If you click it, you will see:

DS1104DAC_Cx



Purpose

To write to one of the 8 parallel D/A converter channels.

I/O mapping

For information on the I/O mapping, refer to DAC Unit.

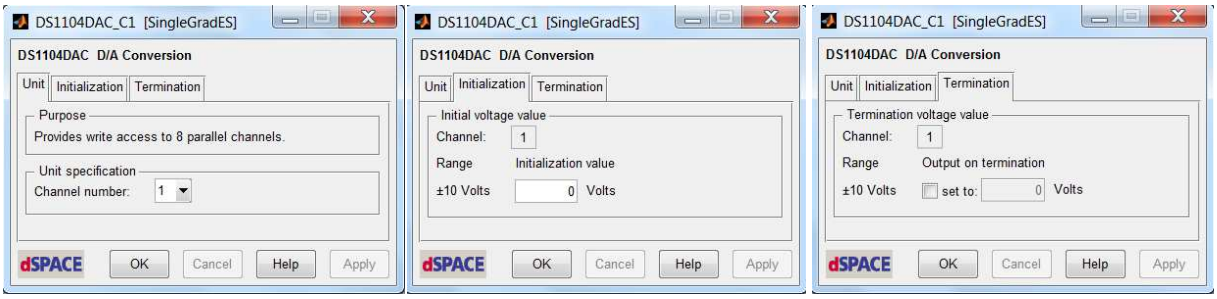
I/O characteristics

- Scaling between the analog output voltage and the input of the block:

Simulink Input	Output Voltage Range
–1 … +1 (double)	–10 V … +10 V

- The block provides its outputs in:
  - Transparent mode, that is the channel is converted and output immediately.
  - Latched mode, that is the channel is converted after synchronous triggering.

Here, note that if you place a DAC block in your Simulink model and double click it there are several settings that need to be made (note the tabs near the top of the window). First, on the *"Unit"* tab you need to select the channel number; here it is channel 1 (DACH1, pin P1A 31). Next, under the *"Initialization" ("Termination")* tab you pick the initial (final) voltage value. Depending on which experiment you hook up, the choice of these values can dictate smooth and safe operation of the experiment (e.g., so that you do not hurt the experimental equipment).

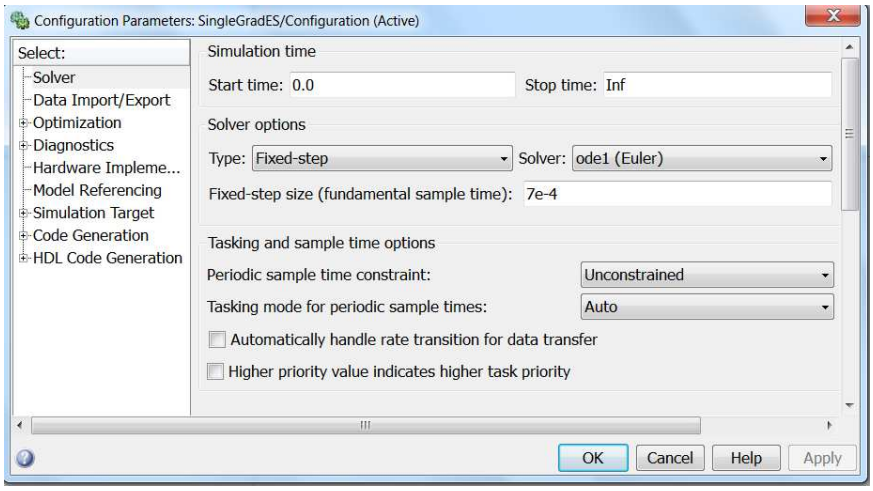


For instance, if the initial value for some mechanical system were 10V, then this may correspond to spinning a motor at its maximum rotational speed. Note that in general these values should be viewed as the ones that are input to the plant immediately before and after the actual control system operates. Hence, for example, if you initialize the output to be zero there may be a sharp change at the first sampling instant when the controller may put out a different value (analogous comments hold for termination).

Note that such a sharp change is something that you may have to pay attention to in an actual implementation since it can have effects on the transient response (e.g., for some experiments you may want to make sure that the initial transients due to such effects have died out before you test the response of the system to a step set point change).

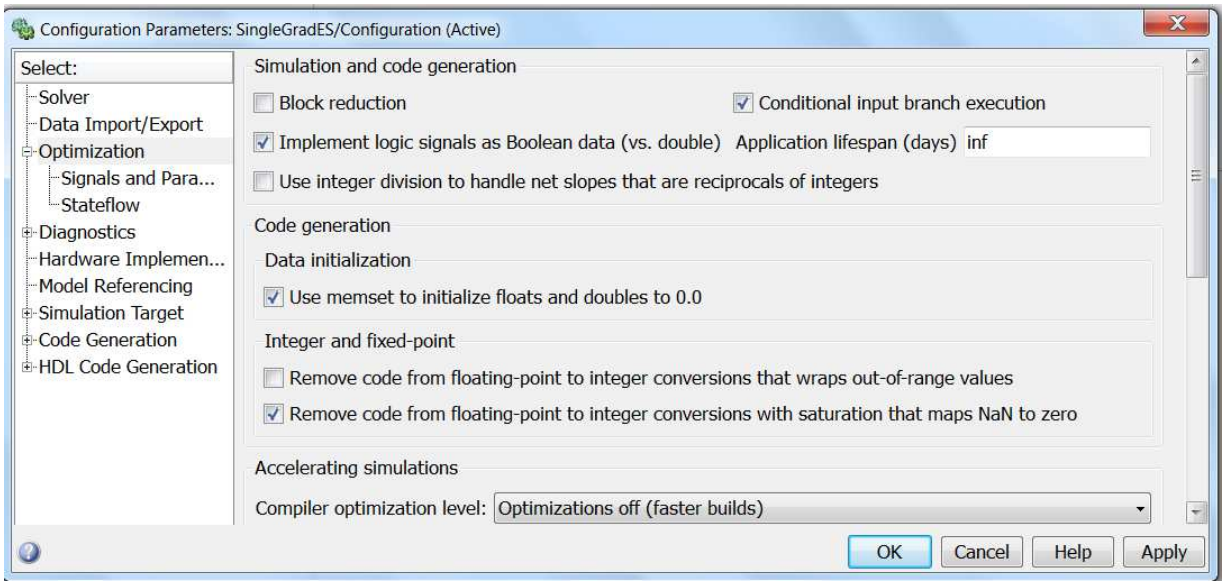
5.3 Building the Simulink Model

Once we define the model, we have to change some parameters in the simulation. To do this, in the Simulink model, use "*Simulation > Configuration Parameters*" and you will see the following window.



First, in the *"Solver"* options (see tab) set the *"Start time"* to 0 (needed for real-time applications). The *"Stop time"* can be set according to how you want the experiment to run. If you set it as *"inf"* it will go forever, but if you set it to 20 it will run the experiment for 20 sec. Next, set the "Type" to a Fixed-step option, and pick a solver such as *"Euler"* or perhaps *"ode5".* Note that the more complex solvers you choose the more computationally intensive your program will be and thus will require more time to execute. Next, pick the sampling time for the experiment. This is the sampling rate, which is typically denoted by "T" in digital control books, and it sets the sampling rate for the sensed signals and control updates. If you have a controller that demands too many computations within the sampling period such that they cannot be completed in time, then you will encounter an overrun condition and you will get an error attesting to this upon download of the program to the DS1104, and you will have to raise the sampling rate.

After you change this, go to the *"Advanced"* option tab, and you should have the *''Block reduction''* option *"Off",* so do that to obtain the next figure:



Once you followed these steps, you are ready to build the model. You have two options: the short-cut command *''CTRL-B''* (from within the Simulink model) or go to *''Tools > Code Generation > Build Model''*. C code is generated for the model and then this code is compiled and linked by the Power PC compiler (since the DS1104 uses a Power PC processor) to produce a single executable object file with a *''.ppc''* extension. This executable is then downloaded to the DS1104 and the program starts running (i.e., executing the controller). If there are any errors during the build process or you run into an overrun condition this will be printed in the MATLAB command window, otherwise if all goes well you will see the message *"Successful completion"* in MATLAB. You can stop the program on the DS1104 in the ControlDesk, on the *''Platform/Device''* tab right click on DS1104 and click on *''Stop RTP''*. Note that stopping the program this way means stopping the whole program, thus the real-time task and the background routine, and that this way will not execute or enable functions associated with the termination state, such as the

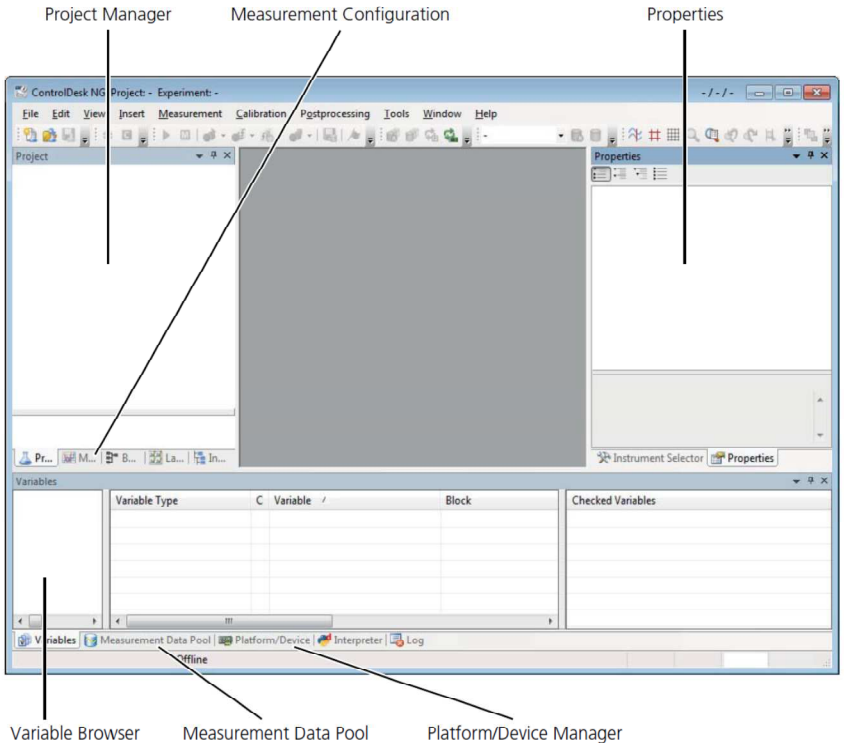
termination values for the DAC channel. To enable the termination condition or state you have to stop only the real-time task, and changing a certain variable in the program does this.

6. Control Desk Environment

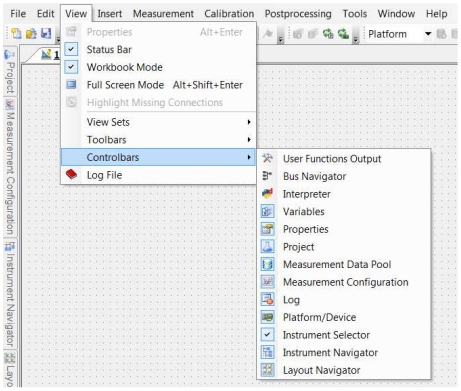
You should sit in front of a computer with dSPACE software and the DS1104 board. Our intent in this first section is to lead you through how to start up the software and understand its main functions. In the next section we will show how to use the software and hardware to implement a very simple control system.

First, from the PC operating system, the following shortcut enables access to the dSPACE ControlDesk environment. If the shortcut does not exist on the desktop please launch ControlDesk from the *"dSPACE ControlDesk 4.2.1"* folder under *"Start\All Programs"*. Either way, once you access it, you will find the following warning message. By checking the box and selecting ''Accept'' button you will not see this message in the future launches.

The following illustration displays ControlDesk's user interface and shows the controlbars you will use in this tutorial.

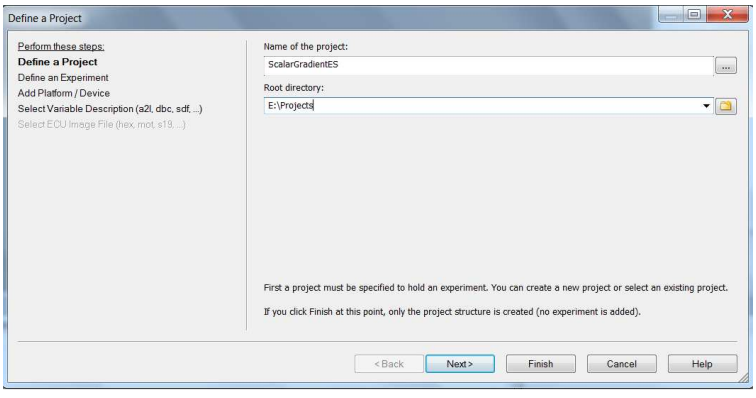


ControlDesk is a user-interface. The DS1104 board is considered a platform on which a simulation is run, just as MATLAB is also a platform to run non-real-time simulations on. From this environment, you will be able to download applications to the DS1104, configure virtual instrumentation that you can use to control, monitor and automate experiments, and develop controllers. Notice that in the view shown above (default window settings for the ControlDesk) you see different regions. *"Platform"* tab shows the different simulation platforms that ControlDesk can interface to. In the region on the bottom (Tool Window), when you select the *"Log Viewer"* tab, you are provided with error and warning messages. The *"Project Manager"* tab presents you with view similar to Windows Explorer as it allows you to browse through the file system of the PC, and choose and download applications with a drag and drop action. The (Python) *"Interpreter"* tab (which uses the "Python" programming language) handles Python commands and scripts for ControlDesk Automation and TestAutomation. Other tabs will appear depending on what you do in the ControlDesk (e.g., when you compile a model as discussed below). The large gray region in the upper middle portion of the screen is a general work area. In this area you can create and display layouts, as well as bring up an editor to write text files, Python scripts or c code. From *''View>Controlbars''* you can hide or show control bars

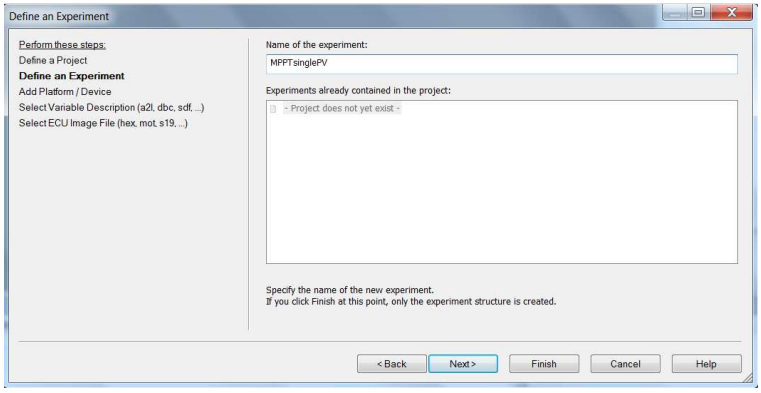


7. How to Prepare the Tutorial Project

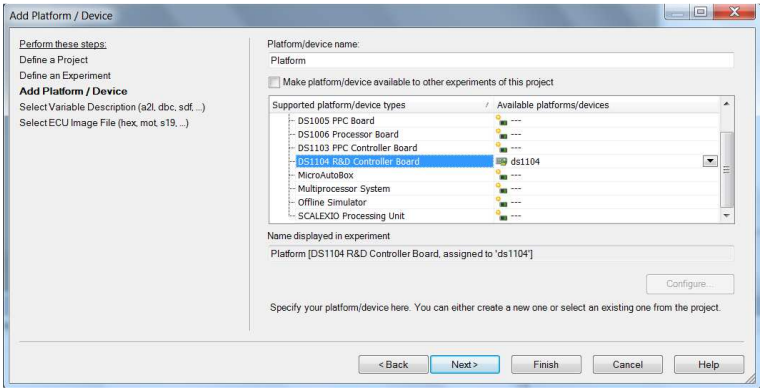
From the *''File menu''*, select *''New - Project+Experiment''*. ControlDesk opens the *''Define a Project''* dialog. In the Name of the project edit field, enter *''GradientScalarES''*.



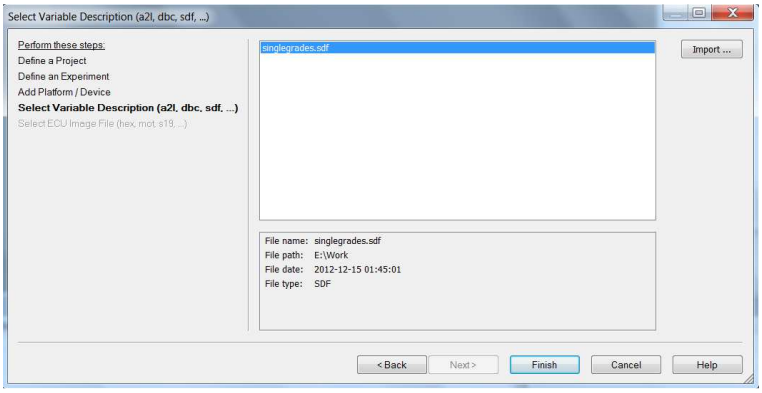
Click *''Next''*. ControlDesk opens the *''Define an Experiment''* dialog. In the Name of the experiment edit field, enter *''MPPTsinglePV''*.



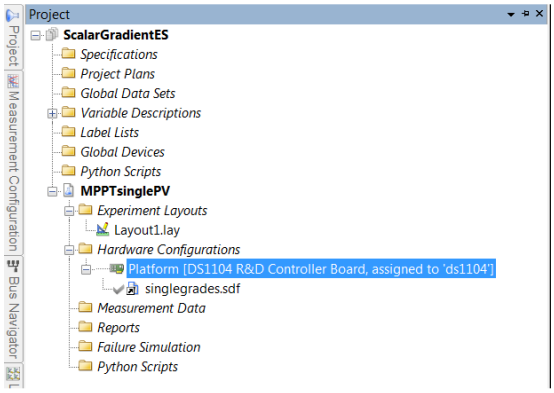
Click *''Next''*. ControlDesk opens the *''Add Platform/Device''* dialog. Select *''DS1104 R&D Control board''*.



Click Import to navigate to the *''E:\Work'' folder* where you have your Simulink files and select the *''singlegrades.sdf''* variable description file for your real-time hardware from the.



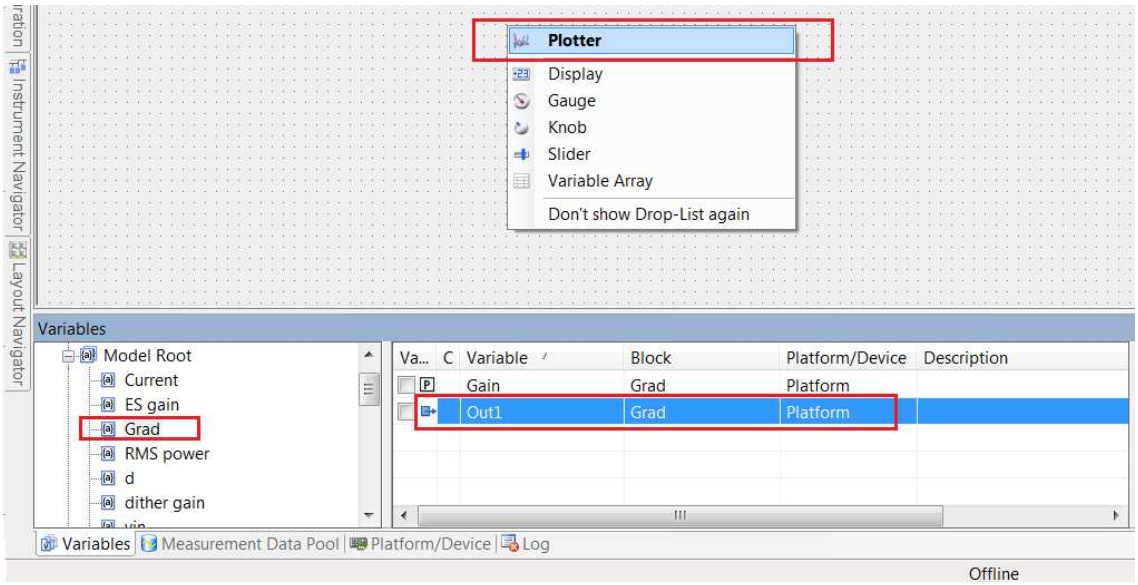
Click *''Finish''*. In the *''Project Manager''*, you can see the project structure you have created so far.



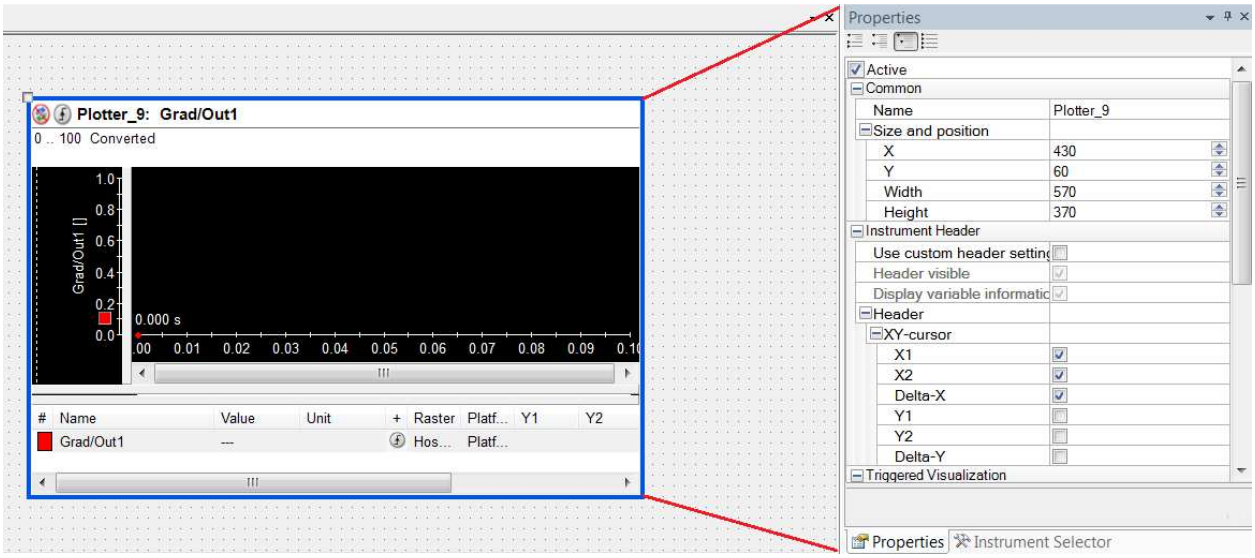
You can define a project with two or more experiments that access the same real-time platform. See ''*Measurement and Recording Tutorial*'' document.

7.1 How to Measure Variable Values

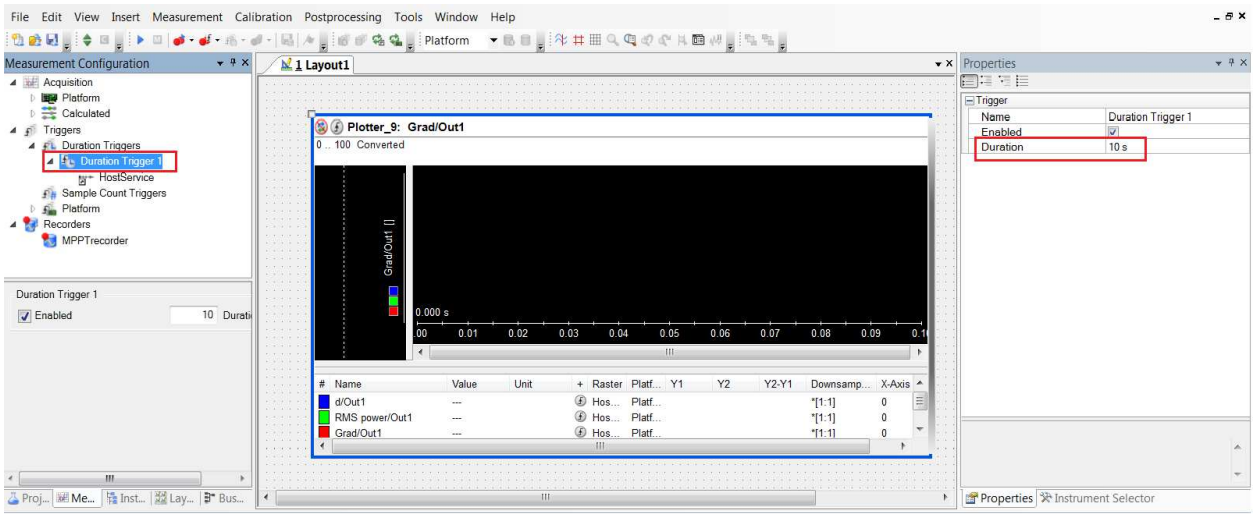
To measure the variable values of a running real-time application, you have to connect an instrument to the variables. A successful measurement also confirms that your ControlDesk Next Generation installation is working correctly. In the tree view of the ''Variable Browser'', navigate to *''singlegrades.sdf > Model Root''*. In the *Variable list*, select the *''Grad > Out1''* variable and drag it to the new layout. In the *''Instrument Type''* list click *''Plotter''*.



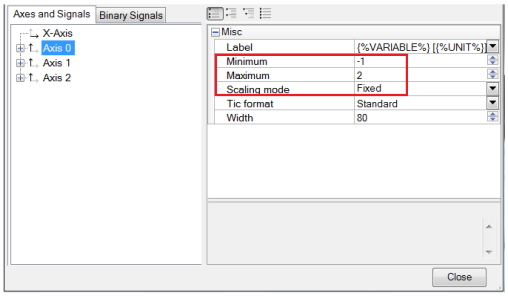
To display or change the properties of the instrument click on the *''Plotter''*. The *''Properties''* control bar shows the Plotter properties.



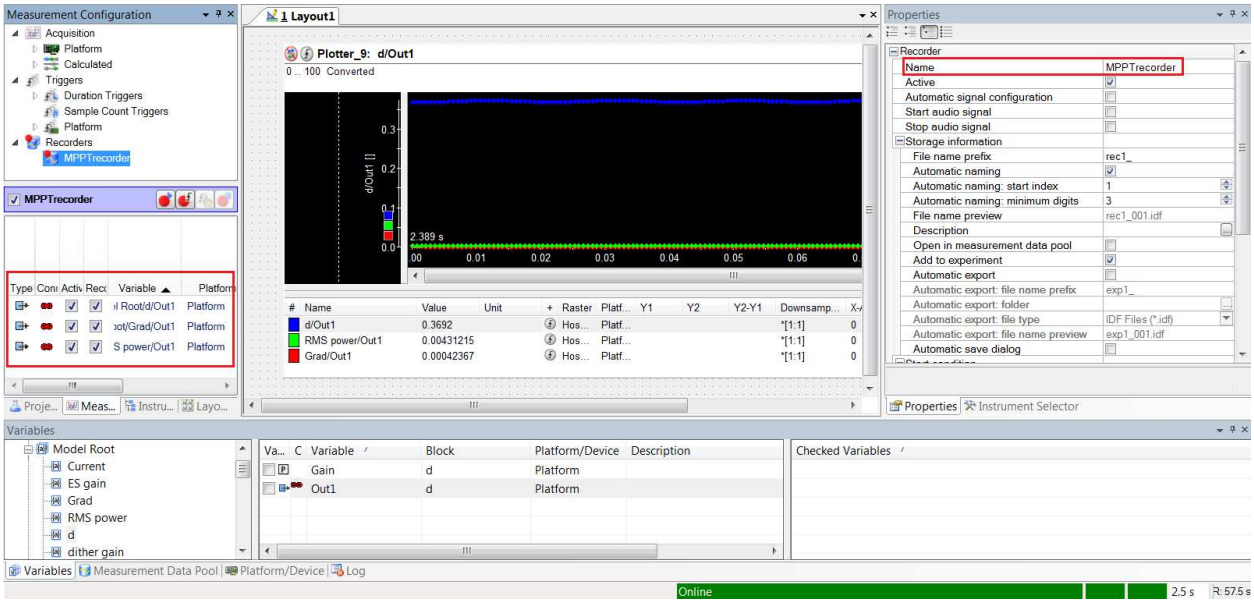
To add another variable to your measurement, change to the RMS power variable group in the tree view of the Variable Browser and select the Out1 variable in the Variable list. Drag it to the Plotter. You can add other variables like *''d > Out1''* to the *Plotter* in the same way. Each measurement is shown in a different color. From the toolbar, select *''Start Measuring''* or press *F5* to measure the variable values.
To expand the time frame shown on the plotter, go to *''Measurement Configuration > Triggers > Duration Trigger1''* and change *''Duration''* to 10s.



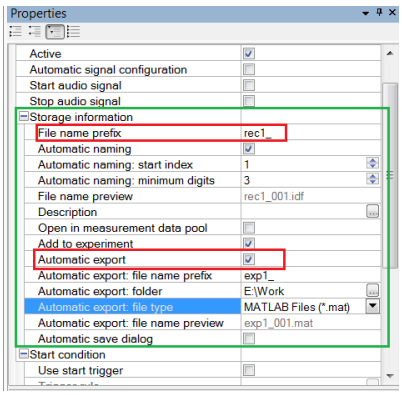
You can change minimum and maximum of the Y-axis from *''Properties''* tab of the plotter. Go to *''Properties > Axes''* and click on the bottom in front of *''Axes''* line. To have a fixed axis widow select *''Fixed''* from *''Scaling mode''* then you can change max and min of the relevant axis.



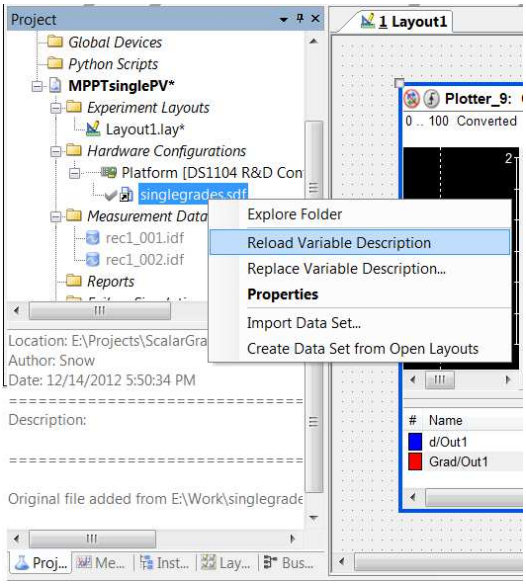
To capture the measurements on the computer hard disk, go to *"Measurement > View Measurement Configuration"* or click *"CTRL+M"*. On the *"Measurement Configuration"* tab go to *"Recorders"*. Right click on the *"Recorders"* then select *"Create New Recorder"*. We name the recorder *"MPPTrecoder"*. Also we can add the variables to the recorder. Drag and drop the desired variables from the *"Variable"* tab to the *"MPPTrecorder"* tab.



You can export the recorded data to your destination folder with *".mat"* extension for future MATLAB uses.



Prior to rebuilding the model in MATLAb or Simulink you shoud stop the online calibration by pushing *"CTRL+F8"*. After you build the new model, go to *"Project > Hardware Configurations"* and right click on "*singlegrades.sdf"* and click on *"Reload Variable Description"*.



8. Experimental Results

Following figures show the results of Maximum Power Point Tracking for a PV module using Extremum Seeking and the DC-DC buck converter.

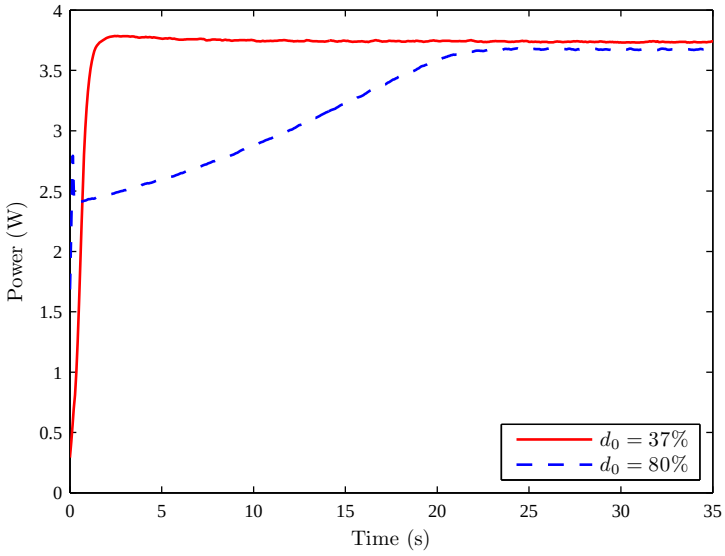


Fig. 9: Power maximization for two different initial conditions. The convergence rate is different on different sides of the power curve of PV module as shown in Fig. 6.

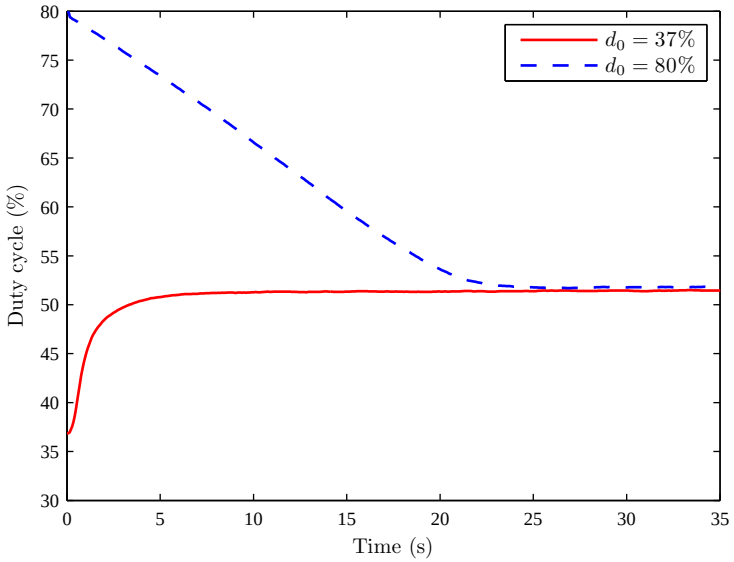


Fig. 10: Variation of duty cycle versus time.

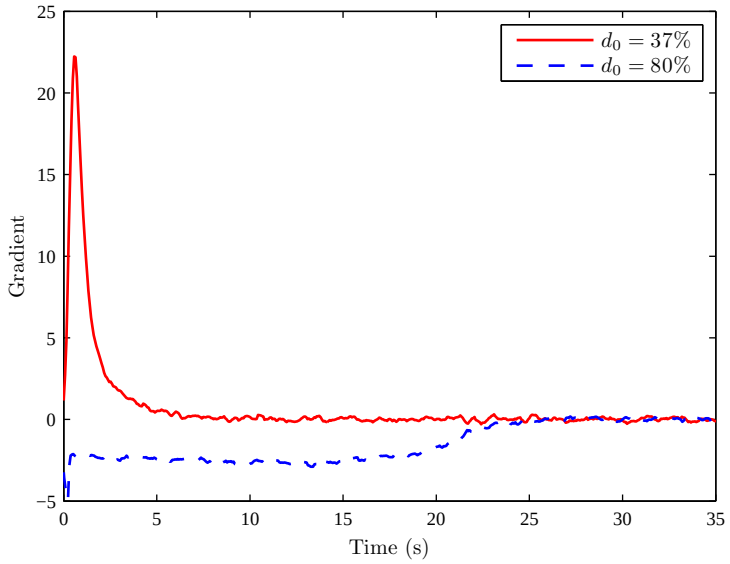


Fig. 11: Variation of the gradient versus time.

9. References

[1] Nicanor Quijano, Kevin Passino, Santhosh Jogi, *"A Tutorial Introduction to Control Systems Development andImplementation with dSPACE"*, Dept. of Electrical Engineering, The Ohio State University, 2002

[2] dSPACE HelpDesk (ControlDesk 4.2.1), dSPACE DVD Release 2012